<https://leetcode.com/problems/two-sum/>

**Two Sum**

**Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.**

**You may assume that each input would have exactly one solution, and you may not use the same element twice.**

**You can return the answer in any order.**

Example 1:

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:

Input: nums = [3,2,4], target = 6

Output: [1,2]

Example 3:

Input: nums = [3,3], target = 6

Output: [0,1]

Constraints:

2 <= nums.length <= 104

-109 <= nums[i] <= 109

-109 <= target <= 109

Only one valid answer exists.

**Method 1:[Brute Force]**

Let key = target – element

For each element in the array find a key in the array except from the current element

Time Complexity: O(n2)  *[ n\*n]*

Space Complexity: O(1) *[constant extra space]*

**Method 2:**

* Sort the array in non-decreasing order.
* Traverse from 0 to N-1
* Initialize searchKey = sum – A[i]
* If(binarySearch(searchKey, A, i + 1, N) == True
* Return True
* Return False

Time Complexity: O(nlogn)  *[ ]*

Space Complexity: O(1) *[constant extra space]*

**Method 3:**

Use a hashmap to store elements and their indices.

For each element check if key is present in hashmap if present return indices

Otherwise insert the current element in the hashmap

Time Complexity: O(nlogn) *[]*

Space Complexity: O(n) *[hashmap ]*

vector<int> twoSum(vector<int>& nums, int target) {

map<int,int> mp;

vector<int> res;

for(int i=0; i<nums.size(); i++){

int key = target - nums[i];

if(mp.count(key)){

res.push\_back(i);

res.push\_back(mp[key]);

}

else

mp[nums[i]] = i;

}

return res;

}